

AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claims 1-25. (canceled)

26. (currently amended) A parser accelerator apparatus having an adaptive learning capability, the apparatus comprising:

a finite state machine configured to parse a document;

a memory configured to store at least one state table;

a token buffer configured to store at least one token received from the finite state machine;

a parser accelerator compiler configured to compile a grammar specification and to generate a state transition specification in a self describing format, the parser accelerator compiler comprising a plurality of object classes including a main class[[,]] and an input manager class having a production rule override section for re-specifying production rules, and further comprising a recursive symbol manager for maintaining a hash table of recursive production rules ~~a token class including an attribute token that describes an allowable occurrence frequency for a symbol in a particular rule for a language;~~ and

a parser accelerator loader configured to load the memory with a state table corresponding to the state transition specification,

wherein the parser accelerator compiler and the parser accelerator loader are configured to reconfigure the parser accelerator in response to detected data patterns

in an electronic document in substantially real time, thereby adapting over time to provide a learning capability.

27. (currently amended) A method for dynamically reconfiguring a parser accelerator comprising:

- providing a grammar specification;
- providing a parser accelerator having a finite state machine and a state table memory;
- compiling the grammar to generate finite automata, the compiling further comprising re-specifying production rules in accordance with a production rules override section of an input manager object class, and maintaining a hash table of recursive production rules using a recursive symbol manager~~using an attribute token of a token object class to describe an allowable occurrence frequency for a symbol in a particular rule for a language;~~
- creating a finite state machine state transition specification in a self describing format from the finite automata; and
- loading the finite state machine state transition specification into the state table memory.

28. (previously presented) The method of claim 27, wherein the self-describing format is a mark-up language.

29. (previously presented) The method of claim 28, wherein the mark-up language is extensible mark-up language (XML).

30. (previously presented) The method of claim 27, wherein the grammar specification includes a specification of desirable performance characteristics of the parser accelerator.

31. (currently amended) A parser accelerator apparatus having an adaptive learning capability, the apparatus comprising:

a finite state machine configured to parse a document;

a memory configured to store at least one state table;

a parser accelerator compiler configured to compile a grammar specification and to generate a state transition specification in a self describing format comprising a mark-up language, the parser accelerator compiler comprising a plurality of object classes including a main class, and an input manager class having a production rule override section for re-specifying production rules, and a token class including an attribute token that describes an allowable occurrence frequency for a symbol in a particular rule for a language

the parser accelerator further comprising a recursive symbol manager for maintaining a hash table of recursive production rules; and

a parser accelerator loader configured to load the memory with a state table corresponding to the state transition specification,

wherein the parser accelerator compiler and the parser accelerator loader are configured to reconfigure the parser accelerator in response to a changed condition.

32. (previously presented) The apparatus of claim 31, wherein the changed condition comprises a pattern in a document.

33. (currently amended) A method for dynamically reconfiguring a parser accelerator comprising:

electronically providing a grammar specification including a specification of desirable performance characteristics of the parser accelerator;

providing a parser accelerator having a finite state machine and a state table memory;

compiling the grammar to generate finite automata, the compiling further comprising re-specifying production rules in accordance with a production rules override section of an input manager object class, and maintaining a hash table of recursive production rules using a recursive symbol manager~~using an attribute token of a token object class to describe an allowable occurrence frequency for a symbol in a particular rule for a language;~~

creating a finite state machine state transition specification from the finite automata; and

loading the finite state machine state transition specification into the state table memory.

34. (previously presented) The method of claim 33, wherein creating a finite state machine state transition specification from the finite automata comprises creating a finite state machine state transition specification in extensible mark-up language (XML).

35. (new) The apparatus of claim 26, wherein the plurality of object classes further comprises a token class including an attribute token that describes an allowable occurrence frequency for a symbol in a particular rule for a language.

36. (new) The method of claim 27, wherein said step of compiling further comprises using an attribute token of a token object class to describe an allowable occurrence frequency for a symbol in a particular rule for a language.

37. (new) The apparatus of claim 31, wherein the plurality of object classes further comprises a token class including an attribute token that describes an allowable occurrence frequency for a symbol in a particular rule for a language.

38. (new) The method of claim 33, wherein said step of compiling further comprises using an attribute token of a token object class to describe an allowable occurrence frequency for a symbol in a particular rule for a language.